Overlapping community detection in labeled graphs

Esther Galbrun, Aristides Gionis, Nikolaj Tatti

Noname manuscript No.
(will be inserted by the editor)

Overlapping community detection in labeled graphs

Esther Galbrun · Aristides Gionis · Nikolaj Tatti

Received: date / Accepted: date

Abstract We present a new approach for the problem of finding overlapping communities in graphs and social networks. Our approach consists of a novel problem definition and three accompanying algorithms. We are particularly interested in graphs that have labels on their vertices, although our methods are also applicable to graphs with no labels.

Our goal is to find k communities so that the total *edge density* over all k communities is maximized. In the case of labeled graphs, we require that each community is succinctly described by a set of labels. This requirement provides a better understanding for the discovered communities. The proposed problem formulation leads to the discovery of vertex-overlapping and dense communities that cover as many graph edges as possible. We capture these properties with a simple objective function, which we solve by adapting efficient approximation algorithms for the generalized maximum-coverage problem and the densest-subgraph problem. Our proposed algorithm is a generic greedy scheme. We experiment with three variants of the scheme, obtained by varying the greedy step of finding a dense subgraph.

We validate our algorithms by comparing with other state-of-the-art community-detection methods on a variety of performance measures. Our experiments confirm that our algorithms achieve results of high quality in terms of the reported measures, and are practical in terms of performance.

Esther Galbrun
Department of Computer Science, Boston University, MA USA
E-mail: galbrun@cs.bu.edu

Aristides Gionis · Nikolaj Tatti
Helsinki Institute for Information Technology (HIIT) and
Department of Information and Computer Science, Aalto University, Finland
E-mail: aristides.gionis@aalto.fi

Nikolaj Tatti
E-mail: nikolaj.tatti@aalto.fi

1 Introduction

As our ability to record and process large amounts of data is growing, datasets increase not only in volume but also in richness and complexity. Reflecting this increasing complexity, the simple graph model is often not adequate to capture many real-world network datasets. For example, for most social networks, information is available not only about social connections but also about user demographics, preferences, actions performed, and so on. Using *attributes* and *labels* for the graph vertices and edges is a natural way to enrich the basic graph data model and capture the additional available information.

Community discovery is one of the most widely-applicable approaches for understanding the structure of networks and many community-detection methods are available. Most methods focus primarily on the graph structure, ignoring other available information. They typically provide a partition of a given graph into disjoint sets of vertices, each set representing a community. The high-level objective is to partition the graph so that there are many edges within communities and few edges across communities, and a large number of different measures has been proposed in attempt to quantify this objective.

On the other hand, many community-detection methods fail to capture (*i*) additional information, such as vertex and edge labels, and (*ii*) overlaps among the graph vertices — detecting overlapping communities is desirable, as in real-world situations each vertex may participate in more than one community. Our goal is to develop community-detection methods that overcome these shortcomings. As we will discuss in our review of the related work, in Section 5, these topics have been the study of other recent research.

Our problem formulation is briefly described as follows. We consider graphs with labels on their vertices. Our model is motivated by social networks, where vertex labels can be used to represent information about individuals, such as occupation, hobbies, preferences, etc. The objective is to discover a set of k communities, so that the total *edge density* over all k communities is maximized. It is also required that each community is succinctly described by a label set S, meaning that the label sets of all the vertices in the community should satisfy a certain property.

Each vertex may belong to more than one community, yielding overlapping communities, but each edge may belong to *at most* one community. Vertices and edges may belong to no community at all — the motivation being that we do not try to artificially force a vertex or an edge into a community. Instead, our k communities should *explain* as much of the graph as possible. An illustration of our setting is shown in Fig. 1.

The constraint that each edge should belong to at most one community is somewhat technical: if we aimed to maximize the total density of the k best communities without this constraint, nothing would prevent us from selecting k almost identical copies of the same (best) community. We would then need a threshold of maximum allowed overlap. By requiring that each edge belongs to at most one community, we circumvent the need for such a threshold while still allowing vertex-overlapping communities. Interestingly, our constraint trans-

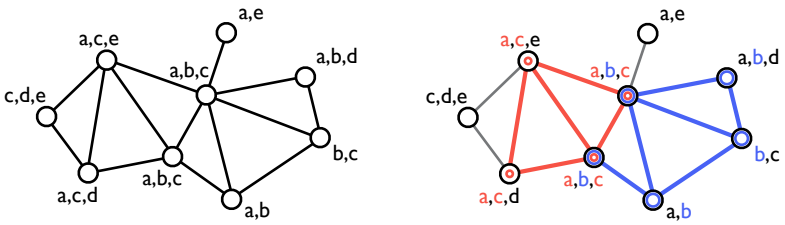


Fig. 1 (Left) Input multi-labeled graph. (Right) Two communities discovered, red and blue, with label sets $\{a, c\}$ and $\{b\}$. Vertices may belong to multiple communities, but each edge is assigned to a single community. Note that not all vertices and all edges need to be assigned to a community.

lates the *density-maximization* objective to an *edge-coverage* objective: the problem becomes equivalent to covering the edges of the input graph with as dense subgraphs as possible.

We also consider the case of non-labeled graphs. In that case, we want to discover k communities with maximum total density. This special case is important as it allows to apply our method to many readily available non-labeled graphs. Furthermore, the solution we develop for the non-labeled case is technically very interesting, and, in fact, we organize our presentation by discussing in depth the non-labeled case. The solution we develop for the non-labeled case directly motivates our algorithms for the labeled graphs.

From the technical side, we show that our problem can be mapped into the *generalized maximum-coverage* problem, for which a greedy algorithm with constant-factor approximation guarantee is available (Cohen and Katzir, 2008). The iterative step of the greedy selects the most cost-effective set, and in our framework this operation translates to finding the densest subgraph. In the case of non-labeled graphs, finding the densest subgraph is a polynomial-time solvable problem, which can also be approximated within a factor of $1/2$ by a linear-time algorithm (Charikar, 2000). Our solution is motivated by this approximation algorithm. In the case of labeled graphs, we propose three algorithms for finding dense subgraphs induced by label sets, which are used in the inner loop of the greedy algorithm. For a simple case of our problem (predicates over small-size label sets) we are able to obtain the same approximation guarantee as the generalized maximum coverage problem.

Our contributions are summarized as follows:

- We propose a new approach for the problem of finding communities in multi-labeled graphs, namely, finding dense overlapping communities that can be succinctly described by label sets.
- We show how to map our community-discovery problem into the generalized maximum-coverage problem (Cohen and Katzir, 2008). Our solution consists of a greedy scheme with three variants for the case of labeled graphs.

– We present a comprehensive empirical evaluation contrasting our algorithms against a variety of state-of-the-art community-detection methods. We show that all three proposed variants achieve results of high quality.

The rest of the paper is organized as follows. In Sections 2 and 3 we introduce our notation and define our problem, respectively, while in Section 4 we present our algorithms. We discuss the related work in Section 5, and we present our empirical evaluation in Section 6. We conclude in Section 7.

2 Notation

We consider a multi-labeled graph $G = (V, E, \ell)$, where V is a set of vertices and E is a set of undirected edges. Labels are associated with the vertices of the graph. We use L to denote the ground set of all possible labels. The function $\ell : V \to 2^L$ is a mapping from vertices to subsets of L, i.e. $\ell(v)$ is the set of labels associated with vertex v. Our goal is to discover tightly-connected communities in the input graph G. To make our problem definition precise we now introduce some notation.

First, given a subset of vertices $U \subseteq V$, we write $E(U)$ for the set of edges of G *induced* by U, that is, $E(U) = \{(x, y) \in E \mid x, y \in U\}$. We will work with subgraphs involving a *subset* of the edges of a full induced subgraph. Namely, we consider subgraphs $H = (U, F)$ with $F \subseteq E(U)$. In other words, while it contains all the vertices in U, H need not contain all the edges induced by it. This is because in our problem, each edge is assigned to *at most one* subgraph. Thus, for a subgraph $H = (U, F)$, some edges in $E(U)$ might not be in F but be assigned to some other subgraph H' instead. The optimal assignment of edges to subgraphs is dictated by our objective function.

Given a subgraph $H = (U, F)$ of the input graph G, we define a reward function $r(H)$ to capture the *quality* of H as a potential community to be output. Perfect cliques are ideal communities. However, in real datasets one cannot expect to find such perfect cliques. Thus, one should be able to cope with noise and look for dense subgraphs. Consequently, we define the reward function to be the *density* of the subgraph $H = (U, F)$, i.e.

$$r(H) = d(H) = \frac{2|F|}{|U|}. \tag{1}$$

The density function $d(H)$ is equal to the *average degree* of H. An alternative density measure is $d'(H) = \frac{2|F|}{|U|(|U|-1)}$, taking values between 0 and 1, where 1 indicates a perfect clique. However, this latter measure does not distinguish between cliques of different sizes, and we use the d measure instead.

We aim to find communities that are not only dense subgraphs, but that can also be described compactly by a set of labels from L. Combining requirements of density of the subgraph and of compactness of the representation by labels leads to better understand the graph structure. We now describe in detail how to use labels in order to define communities.

We first consider 0–1 predicate functions $p : V \times 2^L \to \{0, 1\}$ that for each vertex $v \in V$ and each set of labels $S \subseteq L$ specify whether a certain property holds between the label set $\ell(v)$ and S. Examples of such predicates include:

- *Disjunctive:* $\mathrm{dis}(v, S) = 1$ if $S \cap \ell(v) \neq \emptyset$ and 0 otherwise,
- *Majority:* $\mathrm{maj}(v, S) = 1$ if $\frac{|S \cap \ell(v)|}{|S|} > \frac{1}{2}$ and 0 otherwise,
- *Conjunctive:* $\mathrm{con}(v, S) = 1$ if $S \subseteq \ell(v)$ and 0 otherwise.

Then, a predicate p and a label set S together define a subset of satisfying vertices: $p(S) = \{v \in V \mid p(v, S) = 1\}$. The subset of vertices $p(S)$ that satisfy predicate p is used in the definition of a *label-induced-subgraph*.

Definition 1 (label-induced-subgraph) Let $G = (V, E, \ell)$ be a multi-labeled graph over a label set L, and let $S \subseteq L$ be a set of labels. Let also $U \subseteq V$ and $F \subseteq E$. We say that $H = (U, F)$ is a label-induced-subgraph if the following two properties hold

(*i*) $U = p(S)$, i.e. U is precisely the subset of vertices that satisfy property p;
(*ii*) $F \subseteq E(p(S))$, i.e. the edges of H are a subset of the edges induced by $p(S)$.

Note that our definition does not prohibit a label-induced-subgraph from being disconnected. We indeed allow disconnected subgraphs, but such graphs are naturally penalized by the density reward function. In applications where disconnected subgraphs are not desirable, connectivity constraints can be integrated very easily with our approach.

3 Problem definition

We are now ready to formally define our problem, which we name CoDeL for *community detection in labeled graphs*.

Problem 1 (CoDeL) Let $G = (V, E, \ell)$ be a multi-labeled graph over a set of labels L, let p be a 0–1 predicate over graph vertices and label sets, and let k be a budget parameter. The goal is to find k sets of labels $S_1, \ldots, S_k \subseteq L$, and k *disjoint* sets of edges $F_1, \ldots, F_k \subseteq E$, so that

(*i*) each $H_i = (p(S_i), F_i)$ is a label-induced-subgraph, and
(*ii*) the sum of densities over all the subgraphs H_i

$$d(H_1, \ldots, H_k) = \sum_{i=1}^{k} d(H_i) \tag{2}$$

is maximized. □

The objective stated in Problem 1 is to identify k subgraphs $H_1, \ldots, H_k$ in the multi-labeled graph G so that (*i*) each subgraph H_i is associated with a set of labels S_i, (*ii*) each edge of the input graph G is assigned to *at most one* subgraph H_i, and (*iii*) the sum of densities of the subgraphs H_i is maximized. Thus, we aim at finding a set of dense communities, each described by a set of labels.

We point out that Problem 1 can be interpreted as an *edge-coverage* problem. Indeed, an edge (u, v) assigned to a subgraph $H_i = (p(S_i), F_i)$ with $|p(S_i)| = n_i$ vertices contributes exactly $\frac{2}{n_i}$ to the objective function (2). Since each edge contributes to at most one subgraph, maximizing $d(H_1, \ldots, H_k)$ can be seen as a problem of covering the edges of G with subgraphs.

Notice also that by requiring that each edge is assigned to at most one subgraph, we penalize solutions consisting of nearly identical subgraphs: high-score solutions should cover different parts (edges) of the input graph. On the other hand, our approach allows communities with overlapping vertices. This penalization is needed, otherwise the optimal answer would consists of a single optimal community repeated multiple times. As an alternative approach, one can define a similarity function between the communities and require that the communities should be significantly dissimilar. As a downside, this approach would require to define a similarity between the communities and also provide the threshold.

4 Algorithms

To set up the stage we first define a version of Problem 1 for *non-labeled graphs*, and present a *greedy* algorithm for that problem. Considering the case of non-labeled graphs not only simplifies the exposition of the algorithms for the labeled case, but it is also instructive and interesting in its own sake.

We then proceed to the case of labeled graphs. We present *three* algorithms which follow the same greedy approach and differ only in the inner loop of the greedy step.

4.1 Interlude: non-labeled graphs

We consider the overlapping community-detection problem for *non-labeled graphs*. As before, we want to identify subgraphs $H_1, \ldots, H_k$ in the input graph $G = (V, E)$. Each subgraph H_i is specified by a set of vertices U_i and a set of edges F_i, which is a subset of the edges induced by U_i, i.e. $F_i \subseteq E(U_i)$.

Problem 2 (CoDe) Consider a graph $G = (V, E)$ and a budget parameter k. The goal is to find k disjoint sets of edges $F_1, \ldots, F_k \subseteq E$, and corresponding subgraphs $H_i = (U_i, F_i)$, so that the sum of densities $d(H_1, \ldots, H_k) = \sum_{i=1}^{k} d(H_i)$ is maximized. □

Our algorithm for Problem 2 relies on an approximation algorithm for the *generalized maximum-coverage* problem (GMC) (Cohen and Katzir, 2008), coupled with an approximation algorithm for the *densest-subgraph* problem (DS) (Charikar, 2000). We first present the *generalized maximum-coverage* problem (GMC), as defined by Cohen and Katzir.

Problem 3 (GMC) We are given a set of elements E, a collection of bins B, and a budget k. For each bin $b \in B$, each element $e \in E$ has a positive reward $r(b, e)$ and a non-negative weight $w(b, e)$. In addition, each bin $b \in B$ has a weight $w(b)$ denoting the cost of using that bin. The objective of the GMC problem is to find a set of bins and an assignment of elements to bins in order to maximize the total reward. In particular, a solution to the GMC problem is a triple $C = (X, Y, f)$, where $X \subseteq E$, $Y \subseteq B$, and f is an assignment from X to Y. The total reward r and total weight w of a solution C are

$$r(C) = \sum_{e \in X} r(f(e), e) \quad \text{and} \quad w(C) = \sum_{b \in Y} w(b) + \sum_{e \in X} w(f(e), e). \tag{3}$$

The goal is to find C in order to maximize $r(C)$ subject to $w(C) \leq k$. □

Observation 1 *The* CoDe *problem is an instance of the* GMC *problem.*

Proof Consider an input graph $G = (V, E)$. In order to map CoDe into GMC, we define the elements of E to be the edges of G, and the set of bins B to be the set of all possible subgraphs of G. Given a subgraph $H = (U, F)$, we define the reward of an edge e to be $r(H, e) = \frac{2}{|U|}$. Also, we define $w(H, e) = 0$ if $e \in F$, and finally we define $w(H, e) = \infty$ for $e \notin F$ to make sure that in this case e is never included in the corresponding bin. Finally, we define $w(H) = 1$.

Under this construction, the weight of C in GMC is equal to the number of subgraphs. The budget k corresponds to the number of subgraphs allowed and the total reward is equal to the density $d(H_1, \ldots, H_k)$. □

Observation 1 implies that an approximation algorithm for the GMC algorithm gives an approximation algorithm for the CoDe problem.

Cohen and Katzir provide a greedy algorithm for GMC with approximation guarantee $(\frac{e-1}{2e-1} - \epsilon)$ for every $\epsilon > 0$.[1] Since Observation 1 establishes that CoDe is a special case of the GMC problem, one could naïvely claim that the approximation guarantee for the GMC problem carries over to the CoDe problem. This is not the case, however, because the transformation from CoDe to GMC is not polynomial: The set of all possible subgraphs can be exponential to the input size.

Since the approximation algorithm for the GMC problem is a greedy algorithm, one could potentially overcome the difficulty of having to consider an exponential number of bins. Indeed, for executing a greedy algorithm, it is not necessary to have enumerated all candidate bins (in this case, subgraphs) in advance. It suffices to be able to find the best bin in each iteration of the greedy algorithm. Unfortunately, as we will see, this is a hard problem. In order to understand the problem better, and in order to motivate our suggested algorithm, we first describe briefly the greedy algorithm of Cohen and Katzir.

Approximation algorithm for GMC**.** The algorithm is a variant of the greedy algorithm for the MAX k-COVER problem: Select k bins one by one. In

[1] Cohen and Katzir express their approximation factor as $(\frac{2e-1}{e-1} + \epsilon)$, for every $\epsilon > 0$, but we follow the convention that maximization problems have approx. factors less than 1.

the i-th iteration, select the bin yielding the highest ratio, where the ratio is defined as the ratio of total reward over total weight.

Compared with the standard greedy MAX k-COVER algorithm, the difference of the greedy GMC algorithm arises from the requested mapping of edges to a unique bin. To determine this mapping, the GMC greedy relies on the concept of *residuals*.

Residual rewards of elements are defined as follows. Assume that the current solution is (X, Y, f). That is, element $e \in X$ is assigned to some selected bin $f(e) \in Y$, contributing reward $r(f(e), e)$. In a later iteration, we consider another bin $b' \notin Y$. The residual reward of element e for reassignment $f'(e) = b'$ is the change in reward that would result from reassigning e to b', equal to $r(f'(e), e) - r(f(e), e)$. Intuitively, a positive residual reward indicates that such reassignment would increase the overall reward, improving the solution. The residual reward of an element that has not yet been assigned is defined in a natural way as its original reward.

Residual weights of elements are defined similarly. The residual weight of a bin b is equal to 0 if the bin is part of the current solution, i.e. if $b \in Y$, and equal to its original weight otherwise.

Now, given the current solution, the *residual ratio* obtained by selecting a new bin b' and resetting the edges assignment to f' is the ratio of total rewards to total weights as defined in Equation (3), but considering *residual* rewards and weights in place of the originals.

Then, the greedy algorithm for GMC is the standard greedy algorithm for MAX k-COVER, with the difference that in each iteration the bin yielding the highest *residual* ratio is selected.

Connection with the densest-subgraph problem. Assume for a moment that the greedy GMC algorithm selects the bin yielding the highest ratio (instead of residual ratio) and let us examine the corresponding sub-problem in the context of CoDe. According to our mapping from CoDe to GMC in the proof of Observation 1, we need to pick a subgraph $H_i = (U_i, F_i)$ that maximizes the ratio $d(H_i) = \frac{2|F_i|}{|U_i|}$. In fact, this is a well-studied problem, known as the *densest-subgraph* problem (DS).

Problem 4 (DS) Given a graph $G = (V, E)$, find the subset of vertices $U \subseteq V$ that maximizes the density function $d(U, E(U)) = \frac{2|E(U)|}{|U|}$. □

Although many variants of dense-subgraph problems are **NP**-hard, DS can be solved exactly in polynomial time. Furthermore, there is a linear-time factor-1/2 approximation algorithm (Asahiro et al, 2000; Charikar, 2000), which works as follows. Start with the whole graph and until left with an empty graph, iteratively remove the vertex with the lowest degree (breaking ties arbitrarily) and all its incident edges. Among all subgraphs considered during this vertex-removal process, return the one with the maximum density.

Greedy algorithm for CoDe. A first attempt to adapt the GMC greedy algorithm to the CoDe problem would be the following algorithm: Select k subgraphs iteratively, where the i-th subgraph H_i is found by the linear-time

Algorithm 1: The Dense algorithm

Input: Graph $G = (V, E)$; budget k
Output: Subgraphs $H_1, \ldots, H_k$, with $H_i = (U_i, F_i)$

$W \leftarrow E$; $C \leftarrow \emptyset$ // Uncovered and covered edges
for $i \leftarrow 1$ *to* k **do**
  // Get densest subgraph with residuals
  $H_i = (U_i, F_i) \leftarrow \mathsf{ResidualDensest}(V, W, C, f)$
  $W \leftarrow W \setminus F_i$; $C \leftarrow C \cup F_i$ // Update uncovered and covered edges
  for *all* $e \in F_i$ **do**
    $f(e) \leftarrow i$ // Update assignment of edges to H_i
  end
end
Return $H_1, \ldots, H_k$

densest-subgraph algorithm. For the densest subgraph found, remove its edges from the input graph, and proceed to the $(i+1)$-th step.

The problem with the above algorithm is that it does not account for the opportunity to reassign edges. Assume that an edge $e \in E$ is assigned to some subgraph H_i with n_i vertices, contributing $\frac{2}{n_i}$ to the objective function. At a later step, e might appear in some other subgraph H_j with $n_j < n_i$ vertices. Then, edge e has a positive residual reward, $\frac{2}{n_j} - \frac{2}{n_i}$, meaning that its contribution would increase if reassigned to H_j. Thus, subgraph densities should be computed using residuals and reassigning edges when advantageous.

Unfortunately, unlike the densest-subgraph problem, finding the densest subgraph when considering residuals as defined here is an **NP**-hard problem. We give the precise definition and the proof of this claim in the appendix.

Nevertheless, the above discussion motivates our proposed algorithm, which works as follows. Select iteratively k subgraphs. Let H_i be the subgraph selected in the i-th step, which has n_i vertices. All edges assigned to H_i are *tentatively* removed. However, during the j-th iteration, previously removed edges might be considered again when the size of the subgraph constructed in the execution of Charikar's algorithm drops below the size of the subgraph they are currently assigned to. That is, when the number of vertices remaining in the candidate subgraph becomes n_c, we reinsert edges from any previously selected subgraph H_i such that $n_i > n_c$. Edges contributing to the candidate selected as H_j are assigned to it, including those that have been reinserted during its construction. This reinsertion and reassignment process captures the notion of the residual density, required by the GMC algorithm. Pseudocode for the algorithm, named Dense, is given in Algorithms 1 and 2.

4.2 Finding dense labeled communities

Equipped with the methodology to solve CoDe, we now study the problem for labeled graphs (CoDeL).

We start by pointing out that the CoDeL problem can be solved with the approximation guarantee provided by the GMC greedy algorithm for all

Algorithm 2: The ResidualDensest algorithm

Input: Set of vertices V; uncovered edges W; covered edges C; edge assignment f
Output: Dense subgraph H

$X \leftarrow V$; $R \leftarrow W$ `// Set of vertices and of uncovered edges`
for $j \leftarrow |X|$ *down to* 1 **do**
  $u \leftarrow \text{minDegree}(X, R)$ `// Get the min degree vertex` u `on the graph`(X, R)
  $X \leftarrow X \setminus \{u\}$; $R \leftarrow R \setminus \{(u, z) \in R\}$ `// Remove` u `and all its incident edges`
  `// Reinsert edges` e `from subgraphs with more vertices than the current candidate and having both endpoints in` X
  $R \leftarrow R \cup \left\{\{e \in C \mid \left|V_{f(e)}\right| > |X|\} \cap X \times X\right\}$
  $H_j \leftarrow (X, R)$
end
`// Among all subgraphs considered in the previous iteration, return the one with the highest density`
$H \leftarrow \arg\max_{H_j} d(H_j)$
Return H

predicates that define a polynomial number of subgraphs, as this allows us to perform a single greedy step in a polynomial time. In particular, consider the predicate which holds true if and only if S is a singleton $\{z\}$ with $z \in \ell(v)$, that is, the equivalent restriction to singleton label sets of the three predicates presented in Section 3. Such a predicate generates at most $|L|$ subgraphs. Hence, we can run the GMC greedy algorithm in polynomial time and obtain an approximation guarantee of $(\frac{e-1}{2e-1} - \epsilon)$.

Next, we discuss predicates that may produce an exponential number of subgraphs. In this work, we focus on *conjunctive* predicates. We have also experimented with the *majority* predicate. The results obtained are very similar and we omit the discussion in the interest of conciseness.

We propose *three* algorithms, all based on the greedy approach discussed above. The underlying idea is that when searching for the densest subgraph in the iterative greedy step, instead of considering any dense subgraph, we only consider dense subgraphs that are specified by a label set S. In other words, we start with Dense and replace the subroutine call ResidualDensest with a routine that searches dense subgraphs specified by the labels. The three algorithms differ on how to search for a dense subgraph specified by a label set S. As we will see, they use entirely different strategies.

Algorithm LDense: Greedy on labels. The first algorithm is inspired by Charikar's algorithm for finding densest subgraphs. The difference is that it uses labels to guide the "peeling off" process. In particular, the algorithm removes vertices while maintaining the invariant that at each time there is a label set that specifies the current subgraph.

Specifically, we start with an empty label set $S = \emptyset$ and the corresponding vertex set X_0, containing all the vertices. At the ith step we find a label l such that the corresponding vertex set, $\text{con}(S \cup \{l\})$, has the highest density. Once this is done, we update S and compute the corresponding vertex set, X_i. We

continue until X_i is empty. (Note that adding a new label into S will never increase the size of X_i.) We then choose the densest subgraph among all X_i.

Algorithm Spectral: Spectral ordering on labels. If we could enumerate all $2^{|L|}$ possible label sets, it would be possible to run the greedy with exact sets. Obviously, not all possible label sets give meaningful subgraphs — for instance, most label sets give empty subgraphs. In order to reduce the search space while keeping potentially good label sets we apply the *spectral seriation* method of Atkins et al (1998).

This is done by first forming a similarity matrix M between labels: The similarity score $M[l, l']$ between labels l and l' is equal to the Jaccard coefficient of the sets of vertices that contain label l and that contain label l'. Next, we form the Laplacian of M and order the labels according to the Fielder vector.

We then consider all label sets that correspond to continuous intervals in the above ordering. In the worst case we can have $|L|^2$ such label sets. In practice, however, we obtain much fewer, as only small-sized label sets give non-empty subgraphs.

Algorithm Pivot: Subgraphs pivoted by graph vertices. The third algorithm is inspired by methods that build communities around pivot nodes. For instance, in a recent work by Gupta et al (2014), the authors prove that a social network can be decomposed into dense communities, which are balls of radius 2, centered on pivot vertices.

The Pivot algorithm is an instantiation of the greedy with $|V|$ subgraphs, one for each vertex in V as a pivot. In particular, for each $v \in V$, a subgraph is formed as follows. First all distance-1 neighbors $N_1(v)$ of v are included in the subgraph. Next, each distance-2 neighbor of v is considered and it is included in the subgraph if at least half of its neighbors are in the set $N_1(v)$. We then find a label set S_v that best describes the resulting set of vertices. The label set S_v is constructed in a greedy way, starting from the empty set and adding the best label at each step. Finally, the candidate set of vertices is refined by adding all vertices that satisfy the predicate with S_v and removing those vertices that do not satisfy it.

5 Related work

Community detection is one of the most studied problems in social-network analysis, and a wealth of methods is available. Our review here can by no means be complete; a thorough survey has been produced by Fortunato (2010).

Graph partitioning and disjoint community detection. The majority of the works deal with the problem of partitioning a graph into disjoint communities. A number of different methodologies have been applied, such as hierarchical approaches (Girvan and Newman, 2002), methods based on modularity maximization (Clauset et al, 2004; Girvan and Newman, 2002; White and Smyth, 2005), graph-theoretic approaches (Flake et al, 2000), random-walk methods (Pons and Latapy, 2006; van Dongen, 2000; Zhou and Lipowsky,

2004), label-propagation approaches (van Dongen, 2000), and spectral graph partitioning (Karypis and Kumar, 1998; Ng et al, 2001; von Luxburg, 2007). The above line of research is very different from our work, since we do not aim to partition the graph in disjoint communities, and since our technique is able to handle additional attribute information on the graph vertices.

Overlapping community detection. Researchers in community detection have realized that in many situations it is meaningful to search for overlapping communities. Different methods have been proposed to address this problem, relying on clique percolation (Palla et al, 2005), extensions to the modularity-based approaches (Gregory, 2007; Pinney and Westhead, 2006), analysis of ego-networks (Coscia et al, 2012), game-theoretic approaches (Chen et al, 2010), non-negative matrix factorization (Yang and Leskovec, 2013) or edge clustering (Ahn et al, 2010). A comprehensive survey on the topic of overlapping community detection has been compiled by Xie et al (2011). Our problem formulation makes many novel contributions, with respect to existing work. First, we offer a new combinatorial objective function that aims at partitioning the graph into vertex-overlapping dense subgraphs, while asking to cover as many edges as possible. Second, our approach combines graph structure and vertex labels. As a result, we are able to find communities that not only are dense, but also can be succinctly expressed by a set of labels.

Graphs with label information. Although most methods tend to identify communities using only the graph structure, a number of recent works attempt to combine graph structure with label information. Most of them follow a probabilistic-modeling approach. Balasubramanyan and Cohen (2011) present a model that combines aspects of mixed membership stochastic block models and topic models. Their method improves entity-entity link modeling by jointly modeling links and text about the entities that are linked. In this paper we address a similar problem, following a combinatorial approach. The major differences are identified on how the underlying communities are defined, but also on the techniques used to discover them. Comparative results with the method of Balasubramanyan and Cohen (2011) are included in our experimental evaluation.

McAuley and Leskovec (2012) formulate the problem of automatically identifying social circles in user ego-networks. Their model combines network structure and user profile information, and the approach is based on inferring the parameters of a generative model. Although the problem defined by McAuley and Leskovec (2012) has similar goals as the problem we study in this paper, our experimentation showed that their algorithm performed poorly, both in terms of quality (on the measures we compute), as well as in terms of efficiency (as also remarked by its authors). Thus we do not include the results in this version of our paper.

Descriptive Community Mining was recently introduced by Pool et al (2014) as the problem of mining dense subgraphs that can be described succinctly using vertex labels, specifically disjunctions of (possibly negated) conjunctions. The algorithm proposed by Pool et al. alternates between maximiz-

ing the density of the candidate communities and finding more concise label queries to describe them. Despite the similar problem description, our work differs in some important aspects. In particular, Pool et al. rely on a different measure of density, they adopt a broader description language and use a different strategy for constructing the communities. Results and further discussion of this method are included in our experimental evaluation.

6 Experiments

We evaluate our proposed algorithms, LDense, Pivot and Spectral, by comparing them against a number of baselines on many real-world social networks.

6.1 Datasets

DBLP: The first dataset is the DBLP co-authorship network.[2] Vertices represent researchers, and edges represent the co-authorship relations. Vertex labels represent keywords, extracted from the paper titles in the following way. For each researcher we consider the multi-bag of terms in the titles of his papers. The terms are stemmed and stop-words as well as terms that occur with more than 60% of the authors are removed. For each researcher, we keep the labels whose occurrence count is higher than one percent of the total volume of labels for that researcher.

From the resulting co-authorship network we extract smaller instances. First, we consider some ego-net graphs. We start with 13 high-profile computer scientists,[3] and consider their ego-nets of radius 2. We collectively refer to this collection of ego-nets as DBLP.E2. Second, we consider the subgraphs induced by researchers who have published in the ICDM and KDD conferences, respectively, giving rise to two networks that form the DBLP.C dataset.

G+: The next datasets consist of Google+ ego-nets, kindly distributed via the Stanford network analysis project.[4] This dataset is collected from Google+ users who have shared their circles. We consider two subsets of these networks. The first one, G+.S, contains relatively small ego-nets, that is, graphs with fewer than 42 000 edges and less 300 distinct labels, while the second, G+.L, contains ego-nets with larger numbers of edges and of distinct labels.

FB: This dataset consists of a collection of friend lists from Facebook also made available by the Stanford network analysis project. Vertex labels represent anonymized user attributes. For instance, an attribute "political=Democratic Party" may be anonymized to "political=anonymized feature 1." Overall, there are 10 ego-nets in FB.

[2] http://dblp.uni-trier.de/xml/

[3] Namely, S. Abiteboul, E. Demaine, M. Ester, C. Faloutsos, J. Han, G. Karypis, J. Kleinberg, H. Mannila, K. Mehlhorn, C. Papadimitriou, B. Shneiderman, G. Weikum and P. Yu.

[4] http://snap.stanford.edu

Table 1 Datasets statistics. For each dataset, we indicate how many graphs it consists of ($\#G$), the average number of edges ($|E|$), vertices ($|V|$) and labels ($|L|$), the average density ($d(G)$) and average number of labels per vertex ($|\ell(v)|$) across all graphs in the collection.

Collection	$\#G$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert L\rvert$	$d(G)$	$\lvert\ell(v)\rvert$
DBLP	1	3 461 697	929 937	92 165	7.45	16.54
DBLP.E2	13	6 436	2 197	1 524	5.68	23.32
DBLP.C	2	10 949	3 016	1 365	7.28	14.81
G+.S	43	12 954	472	114	47.31	2.38
G+.L	34	142 139	2 302	506	126.84	3.53
FB	10	8 509	409	223	28.62	6.71
Lastfm	1	12 717	1 892	2 910	13.44	7.30

Lastfm: Finally, our last dataset is a single social network of users from the Last.fm online music system,[5] available on the GroupLens web page.[6]

Statistics on all datasets are provided in Table 1.

6.2 Compared algorithms

The methods proposed here aim to discover communities by exploiting both the vertex labels and the graph structure. Our comparison includes representatives from either end of the spectrum, that is, algorithms using only the vertex labels or only the graph structure. It further includes two recently proposed algorithms that utilize both sources of information.

DBP: This Boolean matrix-factorization algorithm, suggested by Miettinen et al (2008), decomposes the (vertex $\times$ label) matrix and uses the resulting basis vectors to obtain overlapping communities over the graph vertices. It uses only vertex labels and not the graph structure.

Links: The algorithm of Ahn et al (2010) discovers overlapping communities by performing a hierarchical clustering on the edges rather than the vertices of the input graph. Links uses only the graph structure and not vertex labels.

Dense: The greedy algorithm discussed in Section 4 for the CoDe problem. It also uses only the graph structure and not vertex labels.

Block-LDA: Balasubramanyan and Cohen (2011) combine Mixed Membership Stochastic Block models (MMSB) with Latent Dirichelet Allocation (LDA) into a unified model that can be applied to labeled graphs to infer soft-membership of vertices in communities. Thresholding this output then yields hard membership in possibly overlapping communities.

DCM: The algorithm of Pool et al (2014) mines dense subgraphs that can be described succinctly using vertex labels.

LDense, Pivot, and Spectral: These are the three algorithms proposed in this paper, presented in Section 4. They use both the graph structure and the vertex labels to identify overlapping communities, as do Block-LDA and DCM.

[5] http://www.lastfm.com

[6] http://grouplens.org/datasets/hetrec-2011/

We experimented with a number of additional baselines, but for clarity we focus on the most similar and best performing algorithms. Specifically, the Metis algorithm of Karypis and Kumar (1998), and the algorithm of Yan and Gregory (2009) both return a complete partition of the vertices, i.e., they do not allow overlaps, making them less comparable, while the algorithms proposed by Gregory (2007) and by McAuley and Leskovec (2012) completed only on the smallest datasets. The latter method performed rather poorly with respect to our objectives, obtaining very large vertex overlaps (11.87 and 11.96 for `G+.S` and `FB`, respectively), low densities (2.36 and 1.96, resp.) and low label specificities (0.19 and 0.42, resp.), indicating that the goal pursued in this approach is fundamentally different from ours.

Our algorithms are implemented in Python. The code and the datasets are publicly available online.[7] For the other algorithms we experimented with, we used implementations provided by their authors. All experiments, except the ones with DCM, were run on a single core of an 8-core Intel Xeon 2.8 GHz processor with 32 GB of memory and a Linux OS. The experiments with DCM were executed on a 2-core Intel P6200 with 4 GB of memory and a Windows OS.

6.3 Measures and evaluation methodology

The algorithms used in our empirical evaluation are all very different: First, they optimize different criteria. Second, some algorithms take into account vertex labels, while others ignore them. Thus, comparing all these algorithms is tricky, and one has to be very careful with conclusions.

Our evaluation philosophy is to be as comprehensive as possible. We opt for computing a large number of measures that reflect the different characteristics of all the algorithms, and reveal trade-offs. We follow this approach because our proposed algorithms optimize a measure that has not been considered before, and thus we want to contrast this measure with different measures considered by other state-of-the-art algorithms.

The reason why we decided to experiment with algorithms that use only the graph structure or only the vertex labels, and thus, are not directly comparable with our methods, is to explore different angles of the trade-off between the reported measures. For example, Links and Dense, which use only the graph structure and not labels, are expected to give subgraphs that are very dense but not focused in terms of vertex labels. Our algorithms, as well as Block-LDA and DCM, are not expected to give as dense subgraphs as Links and Dense since they enforce constraints on the labels of vertices participating in subgraphs. As we will see, the results we obtain confirm this intuition.

The measures that we compute for all tested algorithms are the following.

1. ls: The *label specificity* measures the extent to which a discovered community can be described by a given set of labels. Given a vertex set $U \subseteq V$ and a label set S, we compute the Jaccard coefficient $J(U, p(S))$, where $p(S)$ is

[7] `http://www.cs.helsinki.fi/u/galbrun/misc/lic/`

the set of vertices $v \in V$ that satisfy $p(v, S)$. The measure ls is the average Jaccard coefficient over all communities found by an algorithm. This measure takes value in the unit interval and assigns greater values to the easily describable communities. In particular, $\text{ls} = 1$ for communities that can be perfectly described by a conjunction of labels.
For the algorithms that do not return a conjunction of labels, namely Dense, Links, Block-LDA and DCM, we construct a posteriori a good candidate set of labels S for each discovered community. We find such a label set with a simple greedy procedure: we first find one single label ℓ that maximizes $J(U, p(\{\ell\}))$, and we keep adding labels greedily as long as this score improves.
On the other hand, each community discovered by DBP and our three proposed algorithms, LDense, Pivot, and Spectral, consists of a vertex set U and a corresponding label set S; thus we can compute the label specificity directly.

Note that by design, we have $\text{ls} = 1$ for our three proposed algorithms.

2. d: We compute the *density* $d(H)$ of each discovered community $H = (U, F)$ as defined by Equation (1). The measure d is then the average density over all communities discovered by an algorithm. Communities should be tightly connected, hence larger values of d are desirable.

3. $|F|$: The *community number of edges* is the average number of edges over all communities discovered by an algorithm.

4. $|U|$: The *community size* is the average number of vertices over all communities discovered by an algorithm.

5. c_E: The *edge coverage* is the fraction of edges belonging to some community.

6. c_V: The *vertex coverage* is the fraction of vertices belonging to at least one community.

7. cut: The *edge cut* is the fraction of edges that have their two end-points in different communities.

8. over: The *vertex overlap* is the average number of communities that each covered vertex belongs to.

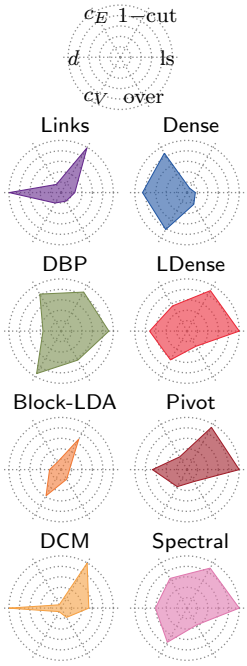
For all measures larger values are favored, except for *edge cut* where smaller is better, and for the vertex overlap which is a neutral measure. By neutral we mean that overlap cannot be an objective to maximize or minimize. Overlap is a property of the data and it is desirable only when there exist overlapping communities in the data. The importance of having large vertex cover depends on the application: if the goal is to explain the whole or almost whole graph with community structure, then we prefer having large vertex cover. On the other hand, if the goal is just find a set of good communities, regardless of their size compared with the whole graph, then the vertex cover plays a smaller role.

6.4 Quantitative results and discussion

We execute each of the algorithms described above with each of our datasets and compute all the measures listed above for each output. The results are reported in Tables 2–7.

Table 2 Quantitative results for the `DBLP.E2` dataset. We report the number of graphs for which results were returned as #. For details on the evaluation measures see Section 6.3. The polar charts visualize the measures for the case $k = 20$.

Algo.	#	ls	d	$\lvert F\rvert$	$\lvert U\rvert$	c_E	c_V	cut	over
				$k = 5$					
Labels only									
DBP	13	0.83	2.57	790	602	0.60	0.76	0.14	1.79
Structure only									
Dense	13	0.20	5.68	729	237	0.54	0.42	0.57	1.22
Links	13	0.25	4.68	99	43	0.08	0.10	0.00	1.03
Labels+Structure									
Block-LDA	13	0.30	3.24	656	411	0.53	0.88	0.34	1.06
DCM	13	0.64	4.04	16	7	0.02	0.02	0.00	1.07
LDense	13	1.00	3.82	549	319	0.39	0.44	0.04	1.51
Pivot	13	1.00	3.64	200	108	0.16	0.20	0.02	1.23
Spectral	13	1.00	3.39	637	388	0.49	0.56	0.06	1.52
				$k = 20$					
Labels only									
DBP	13	0.91	1.38	264	398	0.82	0.94	0.14	3.85
Structure only									
Dense	12	0.16	3.37	252	125	0.87	0.82	0.90	1.55
Links	13	0.26	3.93	55	27	0.18	0.23	0.01	1.13
Labels+Structure									
Block-LDA	13	0.16	0.88	58	78	0.16	0.58	0.32	1.27
DCM	13	0.53	3.96	17	8	0.07	0.08	0.00	1.23
LDense	13	1.00	2.84	187	150	0.56	0.64	0.10	2.02
Pivot	13	1.00	2.63	80	59	0.28	0.38	0.07	1.58
Spectral	13	1.00	2.39	214	189	0.66	0.75	0.11	2.23



To facilitate the analysis, we visualize the results for $k = 20$ with polar charts. The variables in these plots are scaled and we use 1−cut instead of the original cut value, so that for all variables (except overlap, which is a neutral measure) greater values, i.e., further away from the center, indicate better performance.

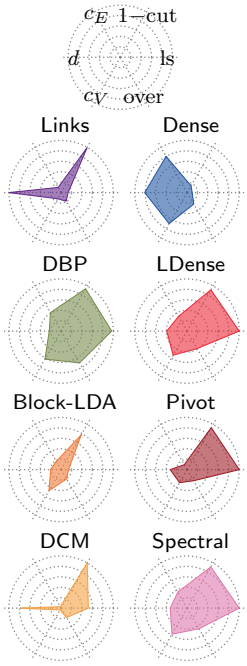
As can be seen from the tables for each algorithm–dataset pair we vary the number of communities k to be returned. DCM produces a ranked list of all communities found in the network from which we selected the top k as the output. All other algorithms take k as input.

Except for `Lastfm`, all other datasets consist of a collection of graphs. The numbers we report are averages over all graphs in a dataset.

Note that the number of graphs for which results are returned (denoted by #) can be smaller than the number of graphs in the collection (denoted by $\#G$ in Table 1). This can happen because of two reasons: First, some algorithms fail to return the required number of clusters, in particular, after covering all vertices with fewer clusters. Second, some algorithms do not scale to the given dataset size, and they do not terminate within a predefined time limit (one day).

Table 3 Quantitative results for the DBLP.C dataset. We report the number of graphs for which results were returned as #. For details on the evaluation measures see Section 6.3. The polar charts visualize the measures for the case $k = 20$.

Algo.	#	ls	d	$\lvert F\rvert$	$\lvert U\rvert$	c_E	c_V	cut	over
				$k = 5$					
Labels only									
DBP	2	1.00	2.25	668	589	0.31	0.53	0.09	1.83
Structure only									
Dense	2	0.12	7.65	1099	262	0.50	0.34	0.54	1.26
Links	2	0.13	5.94	98	34	0.05	0.05	0.00	1.17
Labels+Structure									
Block-LDA	2	0.21	2.69	829	513	0.38	0.80	0.36	1.06
DCM	2	0.58	4.13	17	8	0.01	0.01	0.00	1.42
LDense	2	1.00	2.96	602	421	0.28	0.46	0.08	1.52
Pivot	2	1.00	2.66	118	100	0.05	0.15	0.01	1.23
Spectral	2	1.00	2.81	609	434	0.28	0.47	0.08	1.55
				$k = 20$					
Labels only									
DBP	2	0.97	1.23	223	395	0.41	0.62	0.06	4.22
Structure only									
Dense	2	0.10	4.36	438	157	0.80	0.69	0.85	1.53
Links	2	0.13	5.38	62	23	0.11	0.14	0.01	1.15
Labels+Structure									
Block-LDA	2	0.14	1.07	100	87	0.18	0.47	0.23	1.24
DCM	2	0.52	4.12	17	7	0.03	0.04	0.00	1.25
LDense	2	1.00	2.09	178	186	0.33	0.53	0.09	2.31
Pivot	2	1.00	1.74	58	64	0.11	0.29	0.07	1.48
Spectral	2	1.00	1.73	194	238	0.35	0.58	0.09	2.74



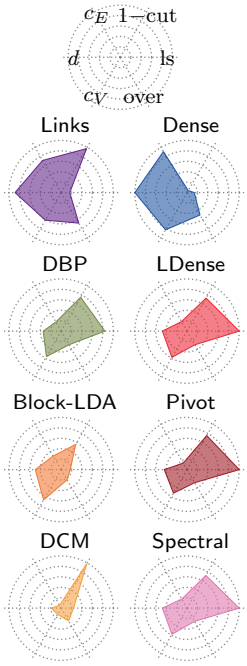
DBP gives the best results in terms of label specificity among the baseline algorithms, but ranks low in terms of graph measures. This is expected, as it is the only baseline algorithm that takes into account only vertex labels and not the graph structure.

On the other hand, the two algorithms that exploit only the graph structure, Dense and Links, give very dense subgraphs at the cost of low label specificity. Note that none of the other algorithms that combines graph structure and content manages to achieve as high densities as these two algorithms. Dense achieves the highest vertex and edge coverage, while in some cases Links gives low values for the coverage measures. In terms of cut score, the performance of Links is excellent, which is quite remarkable given that it does not optimize cut directly. Overall, summarizing the comparison between Dense and Links, we have that Dense gives slightly denser subgraphs and has better vertex and edge coverage, while Links is superior in terms of cut scores.

Block-LDA and DCM, the two baseline algorithms that use both the structure and the label information, rank between these two extremes, with rather strong discrepancies between the different datasets. In particular, DCM achieves some of the top densities on the DBLP datasets but rather poor ones on the G+ datasets while this is reversed for Block-LDA. The communities returned

Table 4 Quantitative results for the G+.S dataset. We report the number of graphs for which results were returned as #. For details on the evaluation measures see Section 6.3. The polar charts visualize the measures for the case $k = 20$.

Algo.	#	ls	d	$\lvert F\rvert$	$\lvert U\rvert$	c_E	c_V	cut	over
				$k = 5$					
Labels only									
DBP	41	0.90	9.36	438	51	0.17	0.42	0.12	1.31
Structure only									
Dense	43	0.17	31.04	1992	94	0.79	0.64	0.83	1.76
Links	43	0.18	25.95	1521	90	0.61	0.51	0.01	2.04
Labels+Structure									
Block-LDA	42	0.15	14.33	835	91	0.31	0.82	0.54	1.12
DCM	13	0.33	8.40	177	19	0.07	0.14	0.01	1.27
LDense	28	1.00	13.38	619	62	0.19	0.43	0.12	1.29
Pivot	28	1.00	13.17	616	63	0.19	0.43	0.12	1.32
Spectral	28	1.00	13.37	621	62	0.19	0.43	0.13	1.28
				$k = 20$					
Labels only									
DBP	35	0.84	4.78	145	25	0.20	0.56	0.25	1.81
Structure only									
Dense	36	0.14	13.91	555	48	0.91	0.82	0.96	2.98
Links	43	0.17	12.29	429	53	0.71	0.61	0.03	4.06
Labels+Structure									
Block-LDA	28	0.17	6.89	260	25	0.30	0.67	0.44	1.36
DCM	3	0.24	2.64	16	5	0.06	0.14	0.02	1.70
LDense	17	1.00	6.56	204	36	0.21	0.58	0.27	1.81
Pivot	12	1.00	6.07	177	38	0.14	0.52	0.25	1.74
Spectral	16	1.00	6.56	208	36	0.20	0.58	0.28	1.75



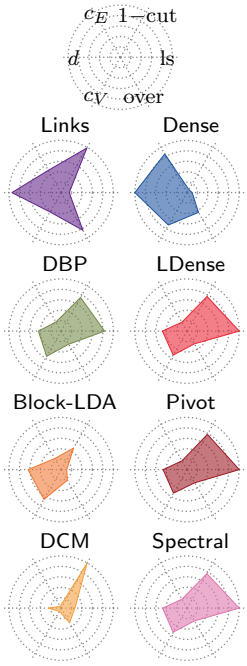
by DCM are notably smaller than those returned by other methods, across all datasets. In particular, in the cases where DCM gives significantly higher densities than the other methods, it returns really small communities.

We now turn to our proposed algorithms, LDense, Spectral and Pivot. Recall that for all three methods the label specificity equals 1 by design. They have similar performances, and can hardly be distinguished on G+. However, LDense returns somewhat denser communities. On the DBLP datasets, the communities returned by Pivot are notably smaller than the ones obtained with the other two methods. While this allows Pivot to achieve higher densities, it also results in poorer coverage measures. Still, the small cuts obtained by the three methods are not an artifact of focusing on small communities.

Overall, compared to Block-LDA and DCM, our algorithms give very good results. First Block-LDA and DCM achieve rather low label specificity. Second, our algorithms consistently give communities that are dense and have low cuts. DCM gives very low cuts and sometimes high densities, however, it returns very small communities. On the other hand, Block-LDA gives sometimes high densities and good coverage scores, but it has poor performance in cut score. The density of the subgraphs returned by our algorithms is of course smaller

Table 5 Quantitative results for the G+.L dataset. We report the number of graphs for which results were returned as #. For details on the evaluation measures see Section 6.3. The polar charts visualize the measures for the case $k = 20$.

Algo.	#	ls	d	$\lvert F\rvert$	$\lvert U\rvert$	c_E	c_V	cut	over
				$k = 5$					
Labels only									
DBP	34	0.88	27.74	5706	256	0.18	0.43	0.12	1.27
Structure only									
Dense	17	0.13	73.32	10977	261	0.69	0.49	0.74	1.64
Links	26	0.17	63.49	10047	287	0.45	0.38	0.01	2.13
Labels+Structure									
Block-LDA	34	0.14	41.29	9959	490	0.34	0.92	0.62	1.18
DCM	20	0.34	18.31	1116	57	0.04	0.07	0.01	1.75
LDense	34	1.00	30.25	5626	232	0.17	0.40	0.11	1.25
Pivot	34	1.00	29.99	5585	233	0.17	0.40	0.11	1.27
Spectral	34	1.00	30.15	5652	235	0.17	0.41	0.11	1.25
				$k = 20$					
Labels only									
DBP	33	0.83	12.43	1594	108	0.20	0.56	0.26	1.74
Structure only									
Dense	11	0.10	28.49	3359	154	0.85	0.72	0.93	2.63
Links	26	0.15	26.89	2756	187	0.49	0.43	0.01	5.00
Labels+Structure									
Block-LDA	33	0.11	18.05	1747	99	0.24	0.66	0.52	1.42
DCM	5	0.26	7.00	381	25	0.04	0.07	0.01	1.93
LDense	34	1.00	13.44	1558	100	0.20	0.53	0.23	1.69
Pivot	34	1.00	13.10	1537	102	0.19	0.52	0.22	1.75
Spectral	34	1.00	13.37	1565	102	0.20	0.53	0.24	1.69



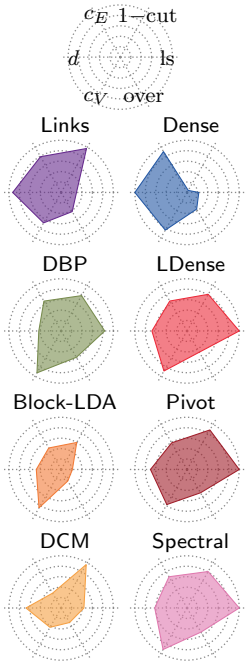
than the densities discovered by Dense and Links, but, as already mentioned, this is because our algorithms are constrained in terms of label specificity.

Running times. The running times of all algorithms with $k = 20$ for the G+ graphs, as a function of the number of edges of the input graph, are shown in Fig. 2. Note the logarithmic scale on both axes. We observe that Dense and Links scale poorly, while LDense and Spectral show the best performance.

To further investigate the scalability of the proposed algorithms, we experiment with the whole DBLP dataset, which has more than 3.4 million edges (see Table 1). The running times are reported in Table 8. Pivot did not complete within 3 days: with that size of a dataset, enumerating individual nodes as candidate pivots is simply infeasible. While LDense was faster than Spectral for 5 communities, Spectral scales much better as a function of the number of communities. The reason is that the label ordering for Spectral needs to be computed only once in the beginning, while the inner loop of LDense for computing densest subgraph candidates is more expensive than the corresponding inner loop of Spectral.

Table 6 Quantitative results for the FB dataset. We report the number of graphs for which results were returned as #. For details on the evaluation measures see Section 6.3. The polar charts visualize the measures for the case $k = 20$.

Algo.	#	ls	d	$\lvert F\rvert$	$\lvert U\rvert$	c_E	c_V	cut	over
				$k = 5$					
Labels only									
DBP	10	0.73	11.65	1120	127	0.61	0.88	0.20	1.81
Structure only									
Dense	10	0.32	24.41	1332	63	0.82	0.70	0.87	1.34
Links	10	0.32	23.42	1138	55	0.66	0.48	0.01	1.63
Labels+Structure									
Block-LDA	10	0.32	14.63	1008	91	0.51	0.91	0.40	1.16
DCM	8	0.46	14.18	676	43	0.25	0.26	0.01	1.67
LDense	10	1.00	14.42	809	77	0.51	0.66	0.10	1.74
Pivot	8	1.00	15.87	890	78	0.42	0.50	0.05	1.89
Spectral	10	1.00	14.04	844	83	0.52	0.67	0.10	1.88
				$k = 20$					
Labels only									
DBP	10	0.84	5.56	309	67	0.66	0.93	0.22	3.50
Structure only									
Dense	8	0.23	12.98	462	40	0.90	0.83	0.95	2.21
Links	10	0.29	12.05	324	31	0.79	0.67	0.03	2.54
Labels+Structure									
Block-LDA	8	0.22	6.18	268	31	0.48	0.85	0.40	1.53
DCM	3	0.44	8.75	260	32	0.33	0.43	0.04	1.93
LDense	8	1.00	8.74	353	54	0.66	0.89	0.19	2.82
Pivot	6	1.00	9.09	405	66	0.59	0.78	0.13	3.10
Spectral	8	1.00	8.04	373	66	0.69	0.92	0.19	3.20



6.5 Case study

We provide a qualitative sample of results generated by the LDense algorithm. For better demonstration of qualitative results, we want to use a more compact dataset, and we extract ego-nets of radius 1 from DBLP. We run the algorithm on two of these ego-nets, the ones of Christos Papadimitriou and Serge Abiteboul. In Tables 9 and 10 we give the top-5 communities extracted for the two authors, respectively. For each community we give the top-10 authors, sorted by their degree in the input graph. We see that: (*i*) the labels used to describe the communities give a very good summary of the research interests of the two authors; (*ii*) for each community, the labels express accurately the common interests of the authors in the community; (*iii*) the overlaps capture the multiple interests of the authors.

7 Conclusions and future work

We presented a new approach for discovering overlapping communities in labeled graphs. We defined the community-detection problem by asking to find k communities whose total edge density is maximized. To provide better un-

Table 7 Quantitative results for the `Lastfm` dataset. We report the number of graphs for which results were returned as #. For details on the evaluation measures see Section 6.3. The polar charts visualize the measures for the case $k = 20$.

Algo.	#	ls	d	$\lvert F\rvert$	$\lvert U\rvert$	c_E	c_V	cut	over
				$k = 5$					
Labels only									
DBP	1	0.21	7.29	1227	310	0.48	0.57	0.08	1.45
Structure only									
Dense	1	0.25	16.78	1185	128	0.47	0.25	0.52	1.38
Links	1	0.31	11.21	345	60	0.14	0.14	0.02	1.09
Labels+Structure									
Block-LDA	1	0.39	8.39	2040	445	0.80	1.00	0.20	1.18
DCM	1	0.51	9.05	109	18	0.04	0.04	0.00	1.26
LDense	1	1.00	8.02	1017	296	0.40	0.27	0.01	2.92
Pivot	1	1.00	7.19	895	253	0.35	0.24	0.02	2.78
Spectral	1	1.00	6.78	1204	393	0.47	0.34	0.02	3.08
				$k = 20$					
Labels only									
DBP	1	0.68	3.45	429	263	0.68	0.83	0.16	3.35
Structure only									
Dense	1	0.18	4.97	557	128	0.88	0.68	0.91	2.00
Links	1	0.23	7.31	131	34	0.21	0.24	0.04	1.53
Labels+Structure									
Block-LDA	1	0.28	3.95	361	116	0.57	0.92	0.34	1.35
DCM	1	0.36	5.18	46	19	0.07	0.08	0.03	2.58
LDense	1	1.00	4.98	280	130	0.44	0.33	0.04	4.17
Pivot	1	1.00	3.90	247	148	0.39	0.28	0.04	5.52
Spectral	1	1.00	4.31	323	194	0.51	0.39	0.03	5.31

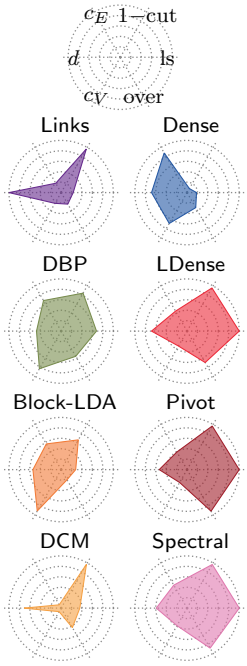


Table 8 Running times of LDense and Spectral on the whole `DBLP` dataset.

Algo.	$k = 5$	$k = 20$	$k = 100$	$k = 1\,000$
LDense	5h 9min	7h 55min	8h 44min	68h 56min
Spectral	6h 35min	6h 44min	7h 24min	12h 57min

derstanding of the discovered communities, we require that each community is succinctly described by a set of labels.

We mapped our density-maximization problem into the generalized maximum-coverage problem, and by adapting the approximation algorithm for the latter problem we obtain three different algorithms, variants of a basic greedy scheme. For a simple case of our problem — predicates over small-size label sets — where we can enumerate all possible label sets, we are able to obtain the same approximation guarantee as the generalized maximum coverage problem.

By conducting thorough experiments on a number of real datasets, where we measured many different performance indicators and compared our algorithms with other state-of-the-art methods, we showed that our algorithms provide consistently some of the best results for most of the measures. In particular, our algorithms are able to discover clusters that, in general, are denser,

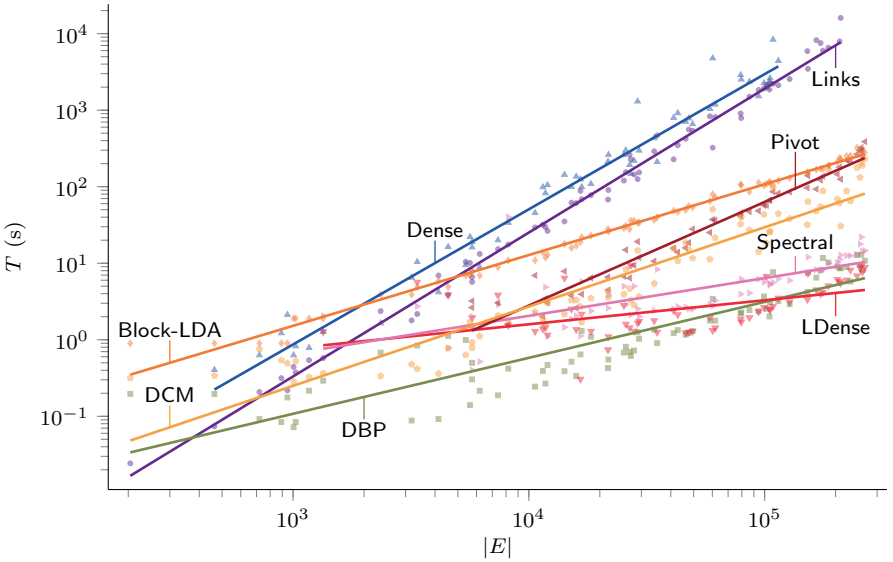


Fig. 2 Running times on G+ networks as a function of the number of edges.

Table 9 Communities for Christos Papadimitriou.

Labels	Authors
logic, query	M. Vardi, G. Kuper, S. Abiteboul, Y. Sagiv, F. Afrati, J. Ullman, P. Kolaitis, R. Fagin, P. Kanellakis, S. Cosmadakis
price	M. Babaioff, R. Kleinberg, N. Immorlica, T. Roughgarden, E. Tardos, A. Mehta, J. Chuang, M. Feldman, V. Vazirani, Y. Singer
mechan, auction	S. Dobzinski, K. Talwar, M. Schapira, R. Sami, A. Archer, M. Babaioff, N. Immorlica, V. Mirrokni, A. Mehta, R. Kleinberg
random	R. Karp, U. Vazirani, V. Vazirani, A. Saberi, M. Blum, A. Wigderson, M. Mihail, O. Goldreich, P. Raghavan, A. Mehta
approxim	M. Yannakakis, K. Jain, X. Deng, C. Daskalakis, A. Kalai, E. Koutsoupias, J. Feigenbaum, M. Sudan, N. Immorlica, P. Goldberg

have smaller cuts, and cover more edges than the communities discovered by the other methods, despite having a more limited search space — they are constrained to communities with label specificity equal to 1. Our experiments also showed that the three proposed algorithms find solutions of similar quality.

Although the emphasis of our approach is on the quality of the communities discovered, we report experiments on a labeled graph with almost a million vertices and more than three million edges, where the Spectral algorithm is shown to be the most efficient. Improving further the scalability of our methods is left for future work.

We restricted ourselves to discovering communities that can be described *exactly* with a label set. This restriction has the virtue that we obtain a description of the whole community. An interesting direction for future work is

Table 10 Communities for Serge Abiteboul.

Labels	Authors
semistructur	J. Wiener, J. McHugh, J. Widom, D. Quass, M. Rys, R. Goldman, S. Chawathe, V. Vassalos, S. Nestorov, T. Lahiri
program	M. Vardi, H. Mairson, P. Kanellakis, C. Beeri, G. Kuper, G. Hillebrand, Y. Sagiv, A. Van Gelder, G. Gottlob, M. Manna
complex	C. Papadimitriou, V. Vianu, E. Waller, S. Grumbach, L. Segoufin, B. Cautis, E. Kharlamov, G. Kuper, G. Gottlob, M. Vardi
xml, web	T. Milo, I. Manolescu, P. Senellart, B. Nguyen, B. Amann, G. Cobena, O. Benjelloun, L. Mignet, A. Marian, N. Preda
xml	I. Manolescu, M. Scholl, S. Cluet, C. Beeri, N. Polyzotis, P. Senellart, T. Milo, V. Vassalos, B. Amann, Y. Papakonstantinou

to relax this constraint and allow communities to have label specificity lower than 1.

From a theoretical perspective, the main open challenge is to devise approximation algorithms that are strongly polynomial in the number of labels in the graph. It will also be interesting to provide approximation guarantees for other cases of our problem definition, such as for non-labeled graphs and other predicate functions.

References

Ahn YY, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. Nature 466:761–764

Asahiro Y, Iwama K, Tamaki H, Tokuyama T (2000) Greedily finding a dense subgraph. Journal of Algorithms 34(2):203–221

Atkins JE, Boman EG, Hendrickson B (1998) A spectral algorithm for seriation and the consecutive ones problem. SIAM Journal on Computing 28:297–310

Balasubramanyan R, Cohen WW (2011) Block-LDA: Jointly modeling entity-annotated text and entity-entity links. In: SIAM International Conference on Data Mining (SDM'11), SIAM / Omnipress, pp 450–461

Charikar M (2000) Greedy approximation algorithms for finding dense components in a graph. International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX'00) pp 84–95

Chen W, Liu Z, Sun X, Wang Y (2010) A game-theoretic framework to identify overlapping communities in social networks. Data Mining and Knowledge Discovery 21(2):224–240

Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Physical Review E p 066111

Cohen R, Katzir L (2008) The generalized maximum coverage problem. Information Processing Letters 108:15–22

Coscia M, Rossetti G, Giannotti F, Pedreschi D (2012) DEMON: a local-first discovery method for overlapping communities. In: Yang Q, Agarwal D, Pei

J (eds) ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12), pp 615–623
van Dongen S (2000) Graph clustering by flow simulation. PhD thesis, University of Utrecht
Flake GW, Lawrence S, Giles CL (2000) Efficient identification of web communities. In: Ramakrishnan R, Stolfo SJ, Bayardo RJ, Parsa I (eds) ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'00), ACM, pp 150–160
Fortunato S (2010) Community detection in graphs. Physics Reports 486
Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99:7821–7826
Gregory S (2007) An algorithm to find overlapping community structure in networks. In: Kok JN, Koronacki J, de Mántaras RL, Matwin S, Mladenic D, Skowron A (eds) European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer, Lecture Notes in Computer Science, vol 4702, pp 91–102
Gupta R, Roughgarden T, Seshadhri C (2014) Decompositions of triangle-dense graphs. In: Naor M (ed) Innovations in Theoretical Computer Science. (ITCS'14), ACM, pp 471–482
Karypis G, Kumar V (1998) Multilevel algorithms for multi-constraint graph partitioning. In: ACM/IEEE Conference on Supercomputing (SC '98), IEEE Computer Society, pp 1–13
von Luxburg U (2007) A tutorial on spectral clustering. Statistics and Computing 17(4):395–416
McAuley J, Leskovec J (2012) Learning to discover social circles in ego networks. In: Bartlett PL, Pereira FCN, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in Neural Information Processing Systems (NIPS'12), pp 548–556
Miettinen P, Mielikäinen T, Gionis A, Das G, Mannila H (2008) The discrete basis problem. IEEE Transactions on Knowledge and Data Engineering, TKDE 20(10):1348–1362
Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: Analysis and an algorithm. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira FCN, Weinberger KQ (eds) Advances in Neural Information Processing Systems (NIPS'01), pp 849–856
Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435:814–818
Pinney J, Westhead D (2006) Betweenness-based decomposition methods for social and biological networks. In: Interdisciplinary Statistics and Bioinformatics, pp 87–90
Pons P, Latapy M (2006) Computing communities in large networks using random walks. Journal of Graph Algorithms Applications 10(2):284–293
Pool S, Bonchi F, van Leeuwen M (2014) Description-Driven Community Detection. ACM Transactions on Intelligent Systems and Technology (ACM TIST) 5(2):1–28

White S, Smyth P (2005) A spectral clustering approach to finding communities in graph. In: SIAM International Conference on Data Mining (SDM'05), SIAM / Omnipress, pp 76–84

Xie J, Kelley S, Szymanski BK (2011) Overlapping community detection in networks: the state of the art and comparative study. arxivorg/abs/11105813

Yan B, Gregory S (2009) Detecting communities in networks by merging cliques. In: IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS'09), pp 832–836

Yang J, Leskovec J (2013) Overlapping community detection at scale: a non-negative matrix factorization approach. In: Leonardi S, Panconesi A, Ferragina P, Gionis A (eds) ACM International Conference on Web Search and Data Mining (WSDM'13), ACM, pp 587–596

Zhou H, Lipowsky R (2004) Network Brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In: Bubak M, Albada G, Sloot P, Dongarra J (eds) Computational Science (ICCS'04), Lecture Notes in Computer Science, vol 3038, pp 1062–1069

Residual dense subgraph is NP-hard

Let us first define the problem of discovering a graph with high residual density.

Problem 5 (ResDenseGraph) Let $G = (V, E, w)$ be a graph with weighted edges. Find a subgraph $H = (X, R)$ such that

$$d(H) - \sum_{e \in R} w(e)$$

is maximized.

Proposition 1 ResDenseGraph *is* ***NP****-hard.*

Proof We will prove hardness by reducing the CLIQUE problem. Assume that we are given a graph $G = (V, E)$ and a size of a clique K. Define the weights to be $w(e) = 2/(2K - 1)$.

Let us assume that G contains a clique of size K, say $H = (X, R)$. We will first show that H has the highest density. To see this let $H' = (X', R')$. Let $N = |X'|$. If $N < K$, then the profit of H' is genuinely smaller than the profit of H. If $N = K$, then the profit of H is larger or equal to the profit of H'. If the profits are equal, then H' has to be a clique as well. Assume that $N > K$. Then we can upper-bound the profit by

$$(N - 1) - N(N - 1)/(2K + 1) = -\frac{(N - 1)(N - 2K + 1)}{2K - 1}.$$

This bound is a parabola, obtaining its apex at K. This shows that the profit of H' is genuinely lower than the profit of H.

We have shown that $G = (V, E)$ has a K-clique if and only if the optimal answer for ResDenseGraph is a clique of size K.